The cyclic-routing UAV problem is PSPACE-complete

HO, Hsi-Ming and OUAKNINE, J.

The Cyclic-Routing UAV Problem is PSPACE-Complete

Hsi-Ming Ho and Joël Ouaknine

Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

Abstract. Consider a finite set of targets, with each target assigned a *relative deadline*, and each pair of targets assigned a fixed transit *flight time*. Given a flock of identical UAVs, can one ensure that every target is repeatedly visited by some UAV at intervals of duration at most the target's relative deadline? The ***Cyclic-Routing UAV Problem*** (CR-UAV) is the question of whether this task has a solution.
This problem can straightforwardly be solved in PSPACE by modelling it as a network of timed automata. The special case of there being a single UAV is claimed to be NP-complete in the literature. In this paper, we show that the CR-UAV Problem is in fact PSPACE-complete even in the single-UAV case.

1 Introduction

Unmanned aerial vehicles (UAVs) have many uses, ranging from civilian to military operations. Like other autonomous systems, they are particularly well-suited to 'dull, dirty, and/or dangerous' missions [21]. A common scenario in such missions is that a set of targets have to be visited by a limited number of UAVs. This has given rise to a large body of research on *path planning* for UAVs.[1] Depending on the specific application at hand, paths of UAVs may be subject to various complex constraints, e.g., related to kinematics or fuel (see, e.g., [1, 17, 19, 23]).

In this work, we consider the *Cyclic-Routing UAV Problem* (CR-UAV) [7]: the decision version of a simple *recurrent* UAV path-planning problem in which each target must be visited not only once but repeatedly, i.e., at intervals of prescribed maximal duration. Problems of this type have long been considered in many other fields such as transportation [16, 22] and robotics [6, 12]. More recently, a number of game-theoretic frameworks have been developed to study similar problems in the context of security [4, 11, 20].

A special case of the problem (with a single UAV) is considered in [3, 4, 13], and is claimed to be NP-complete in [4]. However, the proof of NP-membership in [4] is not detailed.[2] The main result of the present paper is that the CR-UAV Problem is in fact PSPACE-complete, even in the single-UAV case. We note that this problem can be seen as a recurrent variant of the decision version

[1] `http://scholar.google.com/` lists thousands of papers on the subject.
[2] A counterexample to a crucial claim in [4] is given in the full version of this paper [10].

of the *Travelling Salesman Problem with Time Windows* (TSPTW) with upper bounds only (or *TSP with Deadlines* [5]). Its PSPACE-hardness hence stems from recurrence: the decision version of the (non-recurrent) TSPTW Problem is NP-complete [18].

PSPACE-membership of the (general) CR-UAV Problem follows straightforwardly by encoding the problem as the existence of infinite paths in a network of timed automata; we briefly sketch the argument in the next section. The bulk of the paper is then devoted to establishing PSPACE-hardness of the single-UAV case. This is accomplished by reduction from the PERIODIC SAT Problem, known to be PSPACE-complete [15].

2 Preliminaries

2.1 Scenario

Let there be a set of targets and a number of identical UAVs. Each target has a ***relative deadline***: an upper bound requirement on the time between successive visits by UAVs. The UAVs are allowed to fly freely between targets, with a ***flight time*** given for each pair of targets: the amount of time required for a UAV to fly from one of the targets to the other. We assume that flight times are symmetric, that they obey the triangle inequality, and that the flight time from target v to target v' is zero iff v and v' denote the same target. In other words, flight times are a metric on the set of targets. The goal is to decide whether there is a way to coordinate UAVs such that no relative deadline is ever violated. We make a few further assumptions:

- Initially, each UAV starts at some target; there may be more than one UAV at the same target.
- The first visit to each target must take place at the latest by the expiration time of its relative deadline.
- The UAVs are allowed to 'wait' as long as they wish at any given target.
- Time units are chosen so that all relative deadlines and flight times are integers, and moreover all relative deadlines are interpreted as closed constraints (i.e., using non-strict inequalities).

2.2 Modelling via Networks of Timed Automata

We briefly sketch how to model the CR-UAV Problem as the existence of infinite non-Zeno paths in a network of Büchi timed automata, following the notation and results of [2], from which PSPACE-membership immediately follows.

Intuitively, one ascribes a particular timed automaton to each UAV and to each target. Each UAV-automaton keeps track of the location of its associated UAV, and enforces flight times by means of a single clock, which is reset the instant the UAV leaves a given target. Each target-automaton is likewise equipped with a single clock, keeping track of time elapsed since the last visit by some UAV. The action of a UAV visiting a target is modelled by synchronising on a

particular event; when this takes place, provided the target's relative deadline has not been violated, the target resets its internal clock and instantaneously visits a Büchi location. Similarly, the action of a UAV leaving a target is modelled by event synchronisation. Finally, since multiple UAVs may visit a given target simultaneously, each target is in addition equipped with a counter to keep track at any time of whether or not it is currently being visited by some UAV.

The given instance of the CR-UAV Problem therefore has a solution iff there exists a non-Zeno run of the resulting network of timed automata in which each Büchi accepting location is visited infinitely often. By Thm. 7 of [2], this can be decided in PSPACE.

It is worth noting that, since all timing constraints are closed by assumption, standard digitisation results apply (cf. [9]) and it is sufficient to consider integer (i.e., discrete) time. In the next section, we therefore present a discrete graph-based (and timed-automaton independent) formulation of the problem specialised to a single UAV, in order to establish PSPACE-hardness.

2.3 Weighted Graph Formulation

The solution to a single-UAV instance of the CR-UAV Problem consists of an infinite path from target to target in which each target is visited infinitely often, at time intervals never greater than the target's relative deadline. One may clearly assume that the UAV never 'lingers' at any given target, i.e., targets are visited instantaneously. Formally, a single-UAV instance of the CR-UAV Problem can be described as follows. Let V be a set of $n \geq 2$ vertices, with each vertex $v \in V$ assigned a strictly positive integer weight $RD(v)$ (intuitively, the relative deadline of target v). Consider a weighted undirected clique over V, i.e., to each pair of vertices (v, v') with $v \neq v'$, one assigns a strictly positive integer weight $FT(v, v')$ (intuitively, the flight time from v to v'). In addition we require that FT be symmetric and satisfy the triangle inequality.

Let $G = \langle V, RD, FT \rangle$ be an instance of the above data. Given a finite path u in (the clique associated with) G, the ***duration*** $dur(u)$ of u is defined to be the sum of the weights of the edges in u. A ***solution*** to G is an infinite path s through G with the following properties:

- s visits every vertex in V infinitely often;
- Any finite subpath of s that starts and ends at consecutive occurrences of a given vertex v must have duration at most $RD(v)$.

Definition 1 (The CR-UAV Problem with a Single UAV). *Given G as described above, does G have a solution?*

As pointed out in [13], if a solution exists at all then a *periodic* solution can be found, i.e., an infinite path in which the targets are visited repeatedly in the same order.

2.4 The PERIODIC SAT Problem

PERIODIC SAT is one of the many PSPACE-complete problems introduced in [15]. In the following definition (and in the rest of this paper), let $\overline{x}$ be a finite set of variables and let $\overline{x}^j$ be the set of variables obtained from $\overline{x}$ by adding a superscript j to each variable.

Definition 2 (The PERIODIC SAT Problem [15]). *Consider a CNF formula $\varphi(0)$ over $\overline{x}^0 \cup \overline{x}^1$. Let $\varphi(j)$ be the formula obtained from $\varphi(0)$ by replacing all variables $x_i^0 \in \overline{x}^0$ by x_i^j and all variables $x_i^1 \in \overline{x}^1$ by x_i^{j+1}. Is there an assignment of $\bigcup_{j\geq 0} \overline{x}^j$ such that $\bigwedge_{j\geq 0} \varphi(j)$ is satisfied?*

3 PSPACE-Hardness

In this section, we give a reduction from the PERIODIC SAT Problem to the CR-UAV Problem with a single UAV. Consider a CNF formula $\varphi(0) = c_1 \wedge \cdots \wedge c_h$ over $\overline{x}^0 = \{x_1^0, \ldots, x_m^0\}$ and $\overline{x}^1 = \{x_1^1, \ldots, x_m^1\}$. Without loss of generality, we assume that each clause c_j of $\varphi(0)$ is non-trivial (i.e., c_j does not contain both positive and negative occurrences of a variable) and $m > 2$, $h > 0$. We can construct an instance G of the CR-UAV Problem (with the largest constant having magnitude $O(m^2h)$ and $|V| = O(mh)$) such that $\bigwedge_{j\geq 0} \varphi(j)$ is satisfiable if and only if G has a solution.

The general idea of the reduction can be described as follows. We construct *variable gadgets* that can be traversed in two 'directions' (corresponding to assignments **true** and **false** to variables). A *clause vertex* is visited if the corresponding clause is satisfied by the assignment. Crucially, we use *consistency gadgets*, in which we set the relative deadlines of the vertices carefully to ensure that the directions of traversals of the variable gadgets for $\overline{x}^1$ (corresponding to a particular assignment of variables) in a given iteration is consistent with the directions of traversals of the variable gadgets for $\overline{x}^0$ in the next iteration.

3.1 The Construction

We describe and explain each part of G in detail. The reader is advised to glance ahead to Figure 5 to form an impression of G. Note that for ease of presentation, we temporarily relax the requirement that FT be a metric and describe G as an incomplete graph.[3] In what follows, let $l = 24h + 34$ and

$$T = 2\Big(m\big(2(3m+1)l + l\big) + m\big(2(3m+2)l + l\big) + l + 2h\Big).$$

[3] In the single-UAV case, if the FT of some edge is greater than any value in RD, that edge can simply be seen as non-existent.

Variable Gadgets For each variable x_i^0, we construct (as a subgraph of G) a *variable gadget*. It consists of the following vertices (see Figure 1):

- Three vertices on the left side ($LS_i = \{v_i^{t,L}, v_i^{m,L}, v_i^{b,L}\}$)
- Three vertices on the right side ($RS_i = \{v_i^{t,R}, v_i^{m,R}, v_i^{b,R}\}$)
- A '*clause box*' ($CB_i^j = \{v_i^{a,j}, v_i^{b,j}, v_i^{c,j}, v_i^{d,j}, v_i^{e,j}, v_i^{f,j}\}$) for each $j \in \{1, \ldots, h\}$
- A '*separator box*' ($SB_i^j = \{v_i^{\bar{a},j}, v_i^{\bar{b},j}, v_i^{\bar{c},j}, v_i^{\bar{d},j}, v_i^{\bar{e},j}, v_i^{\bar{f},j}\}$) for each $j \in \{0, \ldots, h\}$
- A vertex at the top (v_{top} if $i = 0$, v_{i-1} otherwise)
- A vertex at the bottom (v_i).

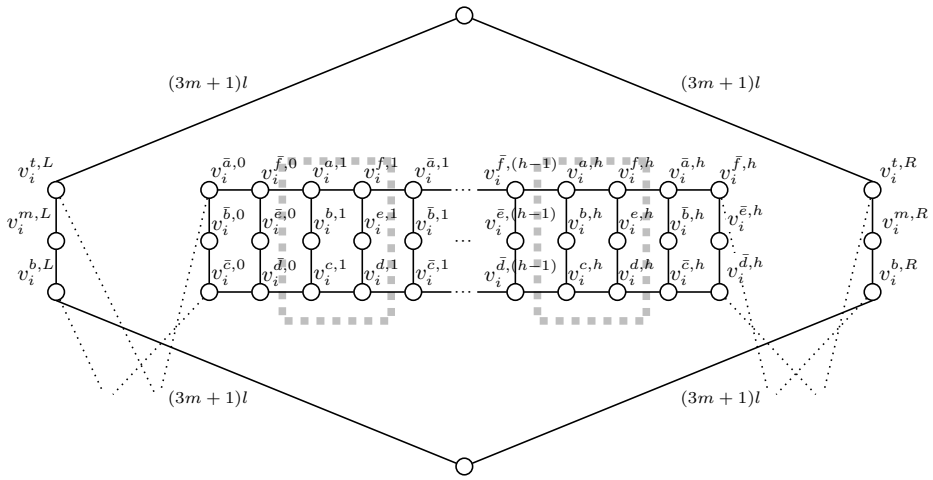

Fig. 1. The variable gadget for x_i^0

The clause boxes for $j \in \{1, \ldots, h\}$ are aligned horizontally in the figure. A separator box is laid between each adjacent pair of clause boxes and at both ends. This row of boxes ($Row_i = \bigcup_{j\in\{1,\ldots,h\}} CB_i^j \cup \bigcup_{j\in\{0,\ldots,h\}} SB_i^j$) is then put between LS_i and RS_i. The RD of all vertices $v \in LS_i \cup RS_i \cup Row_i$ are set to $T + l + 2h$.

The vertices are connected as indicated by solid lines in the figure. The four 'long' edges in the figure have their FT set to $(3m + 1)l$ while all other edges have FT equal to 2, e.g., $FT(v_{top}, v_1^{t,L}) = (3m + 1)l$ and $FT(v_1^{b,1}, v_1^{c,1}) = 2$. There is an exception though: $FT(v_m^{b,L}, v_m)$ and $FT(v_m^{b,R}, v_m)$ (in the variable gadget for x_m^0) are equal to $(3m + 2)l$.

The variable gadgets for variables x_i^1 are constructed almost identically. The three vertices on the left and right side are now LS_{i+m} and RS_{i+m}. The set of vertices in the row is now $Row_{i+m} = \bigcup_{j\in\{1,\ldots,h\}} CB_{i+m}^j \cup \bigcup_{j\in\{0,\ldots,h\}} SB_{i+m}^j$. The vertex at the top is v_{i+m-1} and the vertex at the bottom is v_{i+m} ($i \neq m$) or v_{bot} ($i = m$). The RD of vertices in $LS_{i+m} \cup RS_{i+m} \cup Row_{i+m}$ are set to

$T + l + 2h$, and the FT of the edges are set as before, except that all the 'long' edges now have FT equal to $(3m + 2)l$.

Now consider the following ordering of variables:

$$x_1^0, x_2^0, \ldots, x_m^0, x_1^1, x_2^1, \ldots, x_m^1 .$$

Observe that the variable gadgets for two 'neighbouring' variables (with respect to this ordering) have a vertex in common. To be precise, the set of shared vertices is $S = \{v_1, \ldots, v_{2m-1}\}$. We set the RD of all vertices in S to $T + 2h$ and the RD of v_{top} and v_{bot} to T.

Clause Vertices For each clause c_j in $\varphi(0)$, there is a *clause vertex* v^{c_j} with RD set to $\frac{3}{2}T$. If x_i^0 occurs in c_j as a literal, we connect the j-th clause box in the variable gadget for x_i^0 to v^{c_j} as shown in Figure 2 and set the FT of these new edges to 2 (e.g., $FT(v^{c_j}, v_i^{c,j}) = FT(v^{c_j}, v_i^{d,j}) = 2$). If instead $\neg x_i^0$ occurs in c_j, then v^{c_j} is connected to $v_i^{a,j}$ and $v_i^{f,j}$ (with FT equal to 2). Likewise, the variable gadget for x_i^1 may be connected to v^{c_j} via $\{v_{i+m}^{c,j}, v_{i+m}^{d,j}\}$ (if x_i^1 occurs in c_j) or $\{v_{i+m}^{a,j}, v_{i+m}^{f,j}\}$ (if $\neg x_i^1$ occurs in c_j).

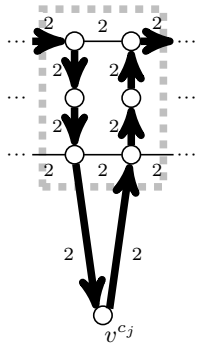

Fig. 2. The variable occurs positively in c_j

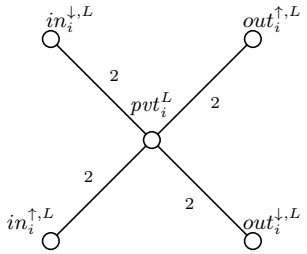

Fig. 3. A consistency gadget LCG_i

Consistency Gadgets For each $i \in \{1, \ldots, m\}$, we construct two *consistency gadgets* LCG_i (see Figure 3) and RCG_i. In LCG_i, the vertex at the centre ($pvt_i^{t,L}$) has RD equal to $\frac{1}{2}T + m\big(2(3m+2)l + l\big) - (2i-1)l + 4h$. The other four vertices ($in_i^{\downarrow,L}$, $out_i^{\uparrow,L}$, $in_i^{\uparrow,L}$ and $out_i^{\downarrow,L}$) have RD equal to $\frac{3}{2}T$. The FT from $pvt_i^{t,L}$ to any of the other four vertices is 2. RCG_i is identical except that the subscripts on the vertices change from L to R.

LCG_i and RCG_i are connected to the variable gadgets for x_i^0 and x_i^1 as in Figure 4. The vertices $in_i^{\downarrow,L}, out_i^{\uparrow,L}, in_i^{\downarrow,R}, out_i^{\uparrow,R}$ are connected to certain vertices in the variable gadget for x_i^0—this allows pvt_i^L and pvt_i^R to be traversed 'from above'. Similarly, the edges connected to $in_i^{\uparrow,L}, out_i^{\downarrow,L}, in_i^{\uparrow,L}, out_i^{\downarrow,L}$ allow pvt_i^L and pvt_i^R to be traversed 'from below'. Formally, $FT(v, v') = 2$ if

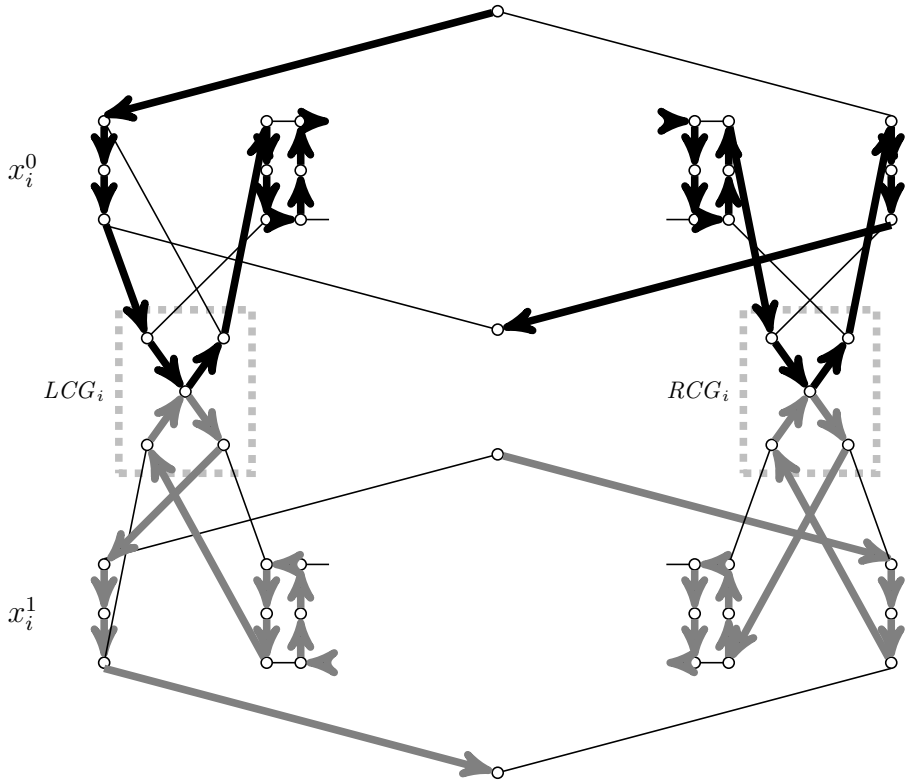

Fig. 4. Connecting the variable gadgets for x_i^0 and x_i^1 to LCG_i and RCG_i

- $v = in_i^{\downarrow,L}, v' \in \{v_i^{b,L}, v_i^{\bar{c},0}\}$ or $v = in_i^{\downarrow,R}, v' \in \{v_i^{\bar{f},h}, v_i^{b,R}\}$
- $v = out_i^{\uparrow,L}, v' \in \{v_i^{t,L}, v_i^{\bar{a},0}\}$ or $v = out_i^{\uparrow,R}, v' \in \{v_i^{\bar{d},h}, v_i^{t,R}\}$
- $v = in_i^{\uparrow,L}, v' \in \{v_{(i+m)}^{b,L}, v_{(i+m)}^{\bar{c},0}\}$ or $v = in_i^{\uparrow,R}, v' \in \{v_{(i+m)}^{\bar{f},h}, v_{(i+m)}^{b,R}\}$
- $v = out_i^{\downarrow,L}, v' \in \{v_{(i+m)}^{t,L}, v_{(i+m)}^{\bar{a},0}\}$ or $v = out_i^{\downarrow,R}, v' \in \{v_{(i+m)}^{\bar{d},h}, v_{(i+m)}^{t,R}\}$.

Two parts of an intended path, which we will explain in more detail later, is also illustrated in Figure 4.

Finally, there is a vertex v_{mid} with $RD(v_{mid}) = T$ connected to v_{bot} and v_{top} with two edges, both with FT equal to $\frac{1}{4}T$. The FT of all the missing edges are $2T$ (note that the largest value in RD is less than $2T$, so these edges can never be taken). This completes the construction of G. An example with $m = 3$ is given in Figure 5, where vertices in S (shared by two variable gadgets) are depicted as solid circles.

The rest of this section is devoted to the proof of the following proposition.

Proposition 3. *$\bigwedge_{j\geq 0} \varphi(j)$ is satisfiable iff G has a solution.*

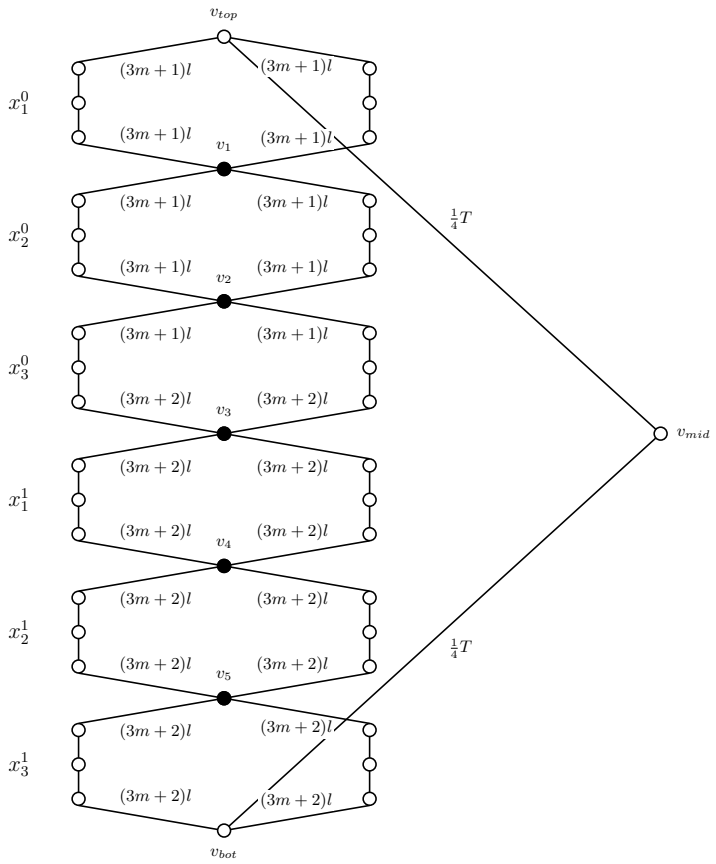

Fig. 5. An example with $m = 3$. Solid circles denote shared vertices $S = \{v_1, \ldots, v_5\}$.

3.2 The Proof of Proposition 3

We first prove the forward direction. Given a satisfying assignment of $\bigwedge_{j\geq 0} \varphi(j)$, we construct a solution s as follows: s starts from v_{top} and goes through the variable gadgets for $x_1^0, x_2^0, \ldots, x_m^0, x_1^1, x_2^1, \ldots, x_m^1$ in order, eventually reaching v_{bot}. Each variable gadget is traversed according to the truth value assigned to its corresponding variable. In such a traversal, both pvt_i^L and pvt_i^R are visited once (see the thick arrows in Figure 4 for the situation when x_i^0 is assigned **true** and x_i^1 is assigned **false**). Along the way from v_{top} to v_{bot}, s detours at certain times and 'hits' each clause vertex exactly once as illustrated by the thick arrows in Figure 2 (this can be done as $\varphi(0)$ is satisfied by the assignment). Then s goes back to v_{top} through v_{mid} and starts over again, this time following the truth values assigned to variables in $\overline{x}^1 \cup \overline{x}^2$, and so on. One can verify that this describes a solution to G.

Now consider the other direction. Let

$$s = (v_{mid} s_1 v_{mid} \dots v_{mid} s_p)^\omega$$

be a periodic solution to G where each *segment* s_j, $j \in \{1, \dots, p\}$ is a finite subpath visiting only vertices in $V \setminus \{v_{mid}\}$. The proofs of the following two propositions can be found in the full version of this paper [10].

Proposition 4. *In $s = (v_{mid} s_1 v_{mid} \dots v_{mid} s_p)^\omega$, either of the following holds:*

- *All s_j, $j \in \{1, \dots, p\}$ starts with v_{top} and ends with v_{bot}*
- *All s_j, $j \in \{1, \dots, p\}$ starts with v_{bot} and ends with v_{top}.*

We therefore further assume that s satisfies the first case of the proposition above (this is sound as a periodic solution can be 'reversed' while remaining a valid solution). We argue that s 'witnesses' a satisfying assignment of $\bigwedge_{j \geq 0} \varphi(j)$.

Proposition 5. *In each segment s_j, each vertex in $\bigcup_{i \in \{1,\dots,m\}} \{pvt_i^L, pvt_i^R\}$ appears twice whereas other vertices in $V \setminus \{v_{mid}\}$ appear once.*

Based on this proposition, we show that s cannot 'jump' between variable gadgets via clause vertices. It follows that the traversal of each Row_i must be done in a single pass.

Proposition 6. *In each segment s_j, if v^{c_k} is entered from a clause box (in some variable gadget), the edge that immediately follows must go back to the same clause box.*

Proof. Consider a 3×3 'box' formed by a separator box and (the left- or right-) half of a clause box. Note that except for the four vertices at the corners, no vertex in this 3×3 box is connected to the rest of the graph. Recall that if each vertex in this 3×3 box is to be visited only once (as enforced by Proposition 5), it must be traversed in the patterns illustrated in Figures 6 and 7.

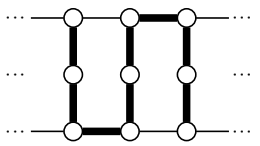

Fig. 6. Pattern '⊔⊓'

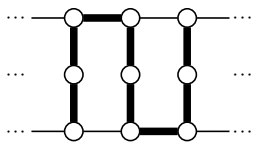

Fig. 7. Pattern '⊓⊔'

Now consider the situation in Figure 8 where s_j goes from v_z to v^{c_k}. The 3×3 box with v_z at its lower-right must be traversed in Pattern '⊔⊓' (as otherwise v_z will be visited twice). Assume that s_j does not visit v_x immediately after v^{c_k}. As v_x cannot be entered or left via v_z and v^{c_k}, the 3×3 box with v_x at its lower-left must also be traversed in Pattern '⊔⊓'. However, there is then no way to enter or leave v_y. This is a contradiction. □

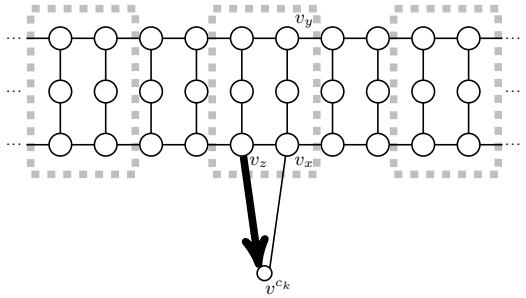

Fig. 8. x_i^0 occurs positively in c_k

Note that in Figure 8, the three clause boxes (framed by dotted lines) are all traversed in Pattern '⊓' or they are all traversed in Pattern '⊔'. More generally, we have the following proposition.

Proposition 7. *In each segment s_j, clause boxes in a given variable gadget are all traversed in Pattern '⊓' or they are all traversed in Pattern '⊔' (with possible detours via clause vertices).*

Write $v \to v'$ for the edge from v to v' and $v \rightsquigarrow v'$ for a finite path that starts with v and ends with v'. By Proposition 5, each segment s_j can be written as $v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_{2m-1}} \rightsquigarrow v_{bot}$ where $b_1, \ldots, b_{2m-1}$ is a permutation of $1, \ldots, 2m-1$. We show that each subpath $v \rightsquigarrow v'$ of s_j with distinct $v, v' \in S \cup \{v_{top}, v_{bot}\}$ and no $v'' \in S \cup \{v_{top}, v_{bot}\}$ in between must be of a very restricted form. For convenience, we call such a subpath $v \rightsquigarrow v'$ a *fragment*.

Proposition 8. *In each segment $s_j = v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_{2m-1}} \rightsquigarrow v_{bot}$, a fragment $v \rightsquigarrow v'$ visits pvt_i^L and pvt_i^R (once for each) for some $i \in \{1, \ldots, m\}$. Moreover, each fragment $v \rightsquigarrow v'$ in $v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_m}$ visits a different set $\{pvt_i^L, pvt_i^R\}$. The same holds for $v_{b_m} \rightsquigarrow v_{b_{m+1}} \rightsquigarrow \cdots \rightsquigarrow v_{bot}$.*

Proof. It is clear that $dur(v \rightsquigarrow v') \geq 2(3m+1)l$, and hence $dur(v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots v_{b_m}) \geq m\big(2(3m+1)l\big)$. Let there be a vertex $v \in \bigcup_{i \in \{1,\ldots,m\}} \{pvt_i^L, pvt_i^R\}$ missing in $v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots v_{b_m}$. Since the time needed from v_{b_m} to v is greater than $(3m+1)l$, even if s_j visits v as soon as possible after v_{b_m}, the duration from v_{bot} in s_{j-1} to v in s_j will still be greater than $\frac{1}{2}T + m\big(2(3m+1)l\big) + (3m+1)l > RD(v)$, which is a contradiction. Therefore, all vertices in $\bigcup_{i \in \{1,\ldots,m\}} \{pvt_i^L, pvt_i^R\}$ must appear in the subpath from v_{top} to v_{b_m}. The same holds for the subpath from v_{b_m} to v_{bot} by similar arguments. Now note that by Proposition 6, a fragment $v \rightsquigarrow v'$ may visit at most two vertices—$\{pvt_i^L, pvt_i^R\}$ for some $i \in \{1, \ldots, m\}$. The proposition then follows from Proposition 5. □

Proposition 9. *In each segment s_j, a fragment $v \rightsquigarrow v'$ visits all vertices in either Row_i or Row_{i+m} for some $i \in \{1, \ldots, m\}$ but not a single vertex in $\bigcup_{j \in \{1,\ldots,m\}} \bigcup_{j \neq i} (Row_j \cup Row_{j+m})$.*

Now consider a fragment $v \rightsquigarrow v'$ that visits pvt_i^L and pvt_i^R (by Proposition 8). By Proposition 5, $v \rightsquigarrow v'$ must also visit exactly two vertices other than pvt_i^L in LCG_i and exactly two vertices other than pvt_i^R in RCG_i (once for each). It is not hard to see that $v \rightsquigarrow v'$ must contain, in order, the following subpaths (together with some obvious choices of edges connecting these subpaths):

(i). A long edge, e.g., $v_i \to v_i^{b,R}$.
(ii). A 'side', e.g., $v_i^{b,R} \to v_i^{m,R} \to v_i^{t,R}$.
(iii). A subpath consisting of a *pvt* vertex and two other vertices in the relevant consistency gadget, e.g., $out_i^{\uparrow,R} \to pvt_i^R \to in_i^{\downarrow,R}$.
(iv). A traversal of a row with detours.
(v). A subpath consisting of a *pvt* vertex and two other vertices in the relevant consistency gadget.
(vi). A side.
(vii). A long edge.

The following proposition is then immediate. In particular, the exact value of $dur(v \rightsquigarrow v')$ is decided by:

- FT of the long edges taken in (i) and (vii)
- detours to clause vertices in (iv).

Proposition 10. *In each segment s_j, the following holds for all fragments $v \rightsquigarrow v'$:*

$$2(3m+1)l + l \leq dur(v \rightsquigarrow v') \leq 2(3m+2)l + l + 2h.$$

Proposition 11. *The order the sets $\{pvt_i^L, pvt_i^R\}$ are visited (regardless of which vertex in the set is first visited) in the first m fragments of each segment s_j is identical to the order they are visited in the last m fragments of s_{j-1}.*

Proof. By Proposition 10, if this does not hold then there must be a *pvt* vertex having two occurrences in s separated by more than $\frac{1}{2}T + m\big(2(3m+1)l + l\big) + 2(3m+1)l$. This is a contradiction. □

For each segment s_j, we denote by $first(s_j)$ the 'first half' of s_j, i.e., the subpath of s_j that consists of the first m fragments of s_j and by $second(s_j)$ the 'second half' of s_j. Write $\exists(v \rightsquigarrow v') \subseteq u$ if u has a subpath of the form $v \rightsquigarrow v'$.

Proposition 12. *In each segment $s_j = v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_{2m-1}} \rightsquigarrow v_{bot}$, we have $b_i = i$ for all $i \in \{1, \ldots, 2m-1\}$.*

Proof. First note that by construction and Proposition 8, $\{pvt_m^L, pvt_m^R\}$ must be the last set of *pvt* vertices visited in $second(s_{j-1})$. By Proposition 11, it must also be the last set of *pvt* vertices visited in $first(s_j)$. Now assume that a long edge of flight time $(3m+2)l$ is taken before pvt_m^L and pvt_m^R are visited in $first(s_j)$. Consider the following cases:

- $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq second(s_{j-1})$ and $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq first(s_j)$: Note that the last edge taken in s_{j-1} is a long edge of flight time $(3m+2)l$, and hence there are two occurrences of pvt_m^L in s separated by at least $\frac{1}{2}T + m\big(2(3m+1)l + l\big) + 2l > \frac{1}{2}T + m\big(2(3m+1)l + l\big) + l + 4h = RD(pvt_m^L)$.

– $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq second(s_{j-1})$ and $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq first(s_j)$: The same argument shows that pvt_m^R must miss its relative deadline.
– $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq second(s_{j-1})$ and $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq first(s_j)$: The same argument shows that both pvt_m^L and pvt_m^R must miss their relative deadlines.
– $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq second(s_{j-1})$ and $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq first(s_j)$: The same argument shows that both pvt_m^L and pvt_m^R must miss their relative deadlines.

We therefore conclude that in $first(s_j)$, all long edges taken before pvt_m^L and pvt_m^R are visited must have FT equal to $(3m+1)l$. Furthermore, all such long edges must be traversed 'downwards' (by Proposition 5). It follows that $b_i = i$ for $i \in \{1, \ldots, m-1\}$. By Proposition 11, Proposition 5 and $m > 2$, we easily derive that $b_m = m$ and then $b_i = i$ for $i \in \{m+1, \ldots, 2m-1\}$. □

By Proposition 12, the long edges in each variable gadget must be traversed in the ways shown in Figures 9 and 10.

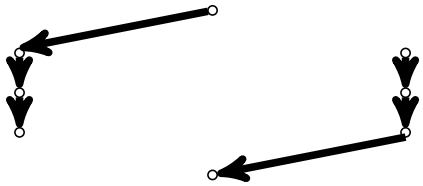

Fig. 9. The variable is assigned to **true**

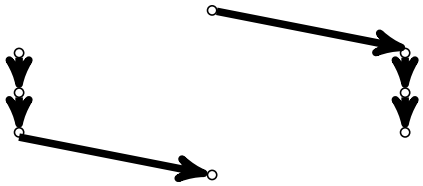

Fig. 10. The variable is assigned to **false**

Proposition 13. *For each segment s_j, the ways in which the long edges are traversed in the last m fragments of s_j are consistent with the ways in which the long edges are traversed in the first m fragments of s_{j+1}.*

Proof. Without loss of generality, consider the case that $\exists(pvt_i^L \rightsquigarrow pvt_i^R) \subseteq second(s_j)$ and $\exists(pvt_i^R \rightsquigarrow pvt_i^L) \subseteq first(s_{j+1})$. By Proposition 12, these two occurrences of pvt_i^L in s are separated by, at least, the sum of $\frac{1}{2}T + m\big(2(3m+2)l + l\big) - (2i-1)l$ and the duration of the actual subpath $pvt_i^R \rightsquigarrow pvt_i^L$ in $first(s_{j+1})$. It is clear that pvt_i^L must miss its relative deadline. □

Proposition 14. *In each segment s_j, if a variable gadget is traversed as in Figure 9 (Figure 10), then all of its clause boxes are traversed in Pattern '⊔' (Pattern '⊓').*

Consider a segment s_j. As each clause vertex is visited once in s_j (by Proposition 5), the ways in which the long edges are traversed in all fragments $v \rightsquigarrow v'$ of s_j (i.e., as in Figure 9 or Figure 10) can be seen as a satisfying assignment of $\varphi(0)$ (by construction and Proposition 14). By the same argument, the ways in which the long edges are traversed in all fragments of s_{j+1} can be seen as a satisfying assignment of $\varphi(1)$. Now by Proposition 13, the assignment of variables $\overline{x}^1$ is consistent in both segments. By IH, s witnesses a (periodic) satisfying assignment of $\bigwedge_{j\geq 0} \varphi(j)$. Proposition 3 is hence proved.

Finally, note that FT can easily be modified into a metric over V by replacing each entry of value $2T$ with the 'shortest distance' between the two relevant vertices. It is easy to see that Proposition 3 still holds. Our main result, which holds for the metric case, follows immediately from Section 2.2.

Theorem 15. *The* CR-UAV *Problem is* PSPACE*-complete.*[4]

4 Conclusion

We have proved that the CR-UAV Problem is PSPACE-complete even in the single-UAV case. The proof reveals a connection between a periodically specified problem and a recurrent path-planning problem (which is not *succinctly specified* in the sense of [14]). We list below some possible directions for future work:

1. A number of crucial problems in other domains, e.g., the generalised pinwheel scheduling problem [8] and the message ferrying problem [24], share similarities with the CR-UAV Problem—namely, they have relative deadlines and therefore 'contexts'. Most of these problems are only known to be NP-hard. It would be interesting to investigate whether our construction can be adapted to establish PSPACE-hardness of these problems.
2. It is claimed in [13] that the restricted case in which vertices can be realised as points in a two-dimensional plane (with discretised distances between points) is NP-complete (with a single UAV). A natural question is the relationship with the problem studied in the present paper.
3. Current approaches to solving the CR-UAV Problem often formulate it as a Mixed-Integer Linear Program (MILP) and then invoke an off-the-shelf solver (see, e.g., [4]). Yet as implied by Proposition 3, the length of a solution can however be exponential in the size of the problem instance. We are currently investigating alternative implementations which would overcome such difficulties.

References

1. Alighanbari, M., Kuwata, Y., How, J.: Coordination and control of multiple uavs with timing constraints and loitering. In: Proceedings of ACC 2003. vol. 6, pp. 5311–5316. IEEE Press (2003)
2. Alur, R.: Timed automata. In: NATO-ASI Summer School on Verification of Digital and Hybrid Systems. Springer (1998), `http://www.cis.upenn.edu/~alur/Nato97.ps`
3. Basilico, N., Gatti, N., Amigoni, F.: Developing a deterministic patrolling strategy for security agents. In: Proceedings of WI-IAT 2009. pp. 565–572. IEEE Computer Society Press (2009)
4. Basilico, N., Gatti, N., Amigoni, F.: Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. Artificial Intelligence 184–185, 78–123 (2012)

[4] Our result holds irrespective of whether the numbers are encoded in unary or binary.

5. Böckenhauer, H.J., Hromkovic, J., Kneis, J., Kupke, J.: The parameterized approximability of TSP with deadlines. Theory Comput. Syst 41(3), 431–444 (2007), http://dx.doi.org/10.1007/s00224-007-1347-x
6. Crama, Y., Van De Klundert, J.: Cyclic scheduling of identical parts in a robotic cell. Operations Research 45(6), 952–965 (1997)
7. Drucker, N., Penn, M., Strichman, O.: Cyclic routing of unmanned air vehicles. Tech. Rep. IE/IS-2014-02, Faculty of Industrial Engineering and Management, Technion (2010), http://ie.technion.ac.il/tech_reports/1393234936_AUVSI-Abstract-31Aug2010-submitted.pdf
8. Feinberg, E.A., Curry, M.T.: Generalized pinwheel problem. Mathematical Methods of Operations Research 62(1), 99–122 (2005)
9. Henzinger, T.A., Manna, Z., Pnueli, A.: What good are digital clocks? In: Proceedings of ICALP 1992. LNCS, vol. 623, pp. 545–558. Springer (1992)
10. Ho, H.M., Ouaknine, J.: The cyclic-routing UAV problem is PSPACE-complete. CoRR abs/1411.2874 (2014), http://arxiv.org/abs/1411.2874
11. Jain, M., Kardes, E., Kiekintveld, C., Ordóñez, F., Tambe, M.: Security games with arbitrary schedules: A branch and price approach. In: Proceedings of AAAI 2010. pp. 792–797. AAAI Press (2010)
12. Kats, V., Levner, E.: Minimizing the number of robots to meet a given cyclic schedule. Annals of Operations Research 69, 209–226 (1997)
13. Las Fargeas, J., Hyun, B., Kabamba, P., Girard, A.: Persistent visitation under revisit constraints. In: Proceedings of ICUAS 2013. pp. 952–957. IEEE Press (2013)
14. Marathe, M.V., Hunt, III, H.B., Stearns, R.E., Radkakrishnan, V.: Complexity of hierarchically and 1-dimensional periodically specified problems. In: Satisfiability Problem: Theory and Applications. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 35, pp. 225–260. DIMACS (1997)
15. Orlin, J.B.: The complexity of dynamic languages and dynamic optimization problems. In: Proceedings of STOC 1981. pp. 218–227. ACM Press (1981)
16. Orlin, J.B.: Minimizing the number of vehicles to meet a fixed periodic schedule: An application of periodic posets. Operations Research 30(4), 760–776 (1982)
17. Richards, A., How, J.P.: Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In: Proceedings of ACC 2002. vol. 3, pp. 1936–1941. IEEE Press (2002)
18. Savelsbergh, M.W.: Local search in routing problems with time windows. Annals of Operations Research 4(1), 285–305 (1985)
19. Sundar, K., Rathinam, S.: Route planning algorithms for unmanned aerial vehicles with refueling constraints. In: Proceedings of ACC 2012. pp. 3266–3271. IEEE Press (2012)
20. Tsai, J., Kiekintveld, C., Ordonez, F., Tambe, M., Rathi, S.: IRIS—a tool for strategic security allocation in transportation networks. In: Tambe, M. (ed.) Security and game theory: algorithms, deployed systems, lessons learned. Cambridge University Press (2009)
21. Unmanned air vehicle systems association. http://www.uavs.org/
22. Wollmer, R.D.: An airline tail routing algorithm for periodic schedules. Networks 20(1), 49–54 (1990)
23. Yang, G., Kapila, V.: Optimal path planning for unmanned air vehicles with kinematic and tactical constraints. In: Proceedings of CDC 2002. vol. 2, pp. 1301–1306. IEEE Press (2002)
24. Zhao, W., Ammar, M.H., Zegura, E.W.: A message ferrying approach for data delivery in sparse mobile ad hoc networks. In: Proceedings of MobiHoc 2004. pp. 187–198. ACM Press (2004)